I DON'T CARE

i am yangqi

contents

1 nlp-blog

关键词：
自然语言处理（NLP）.词向量（Word Vectors）.奇异值分解(Singular Value Decomposition). Skip-gram. 连续词袋（CBOW）,负采样样本（Negative Sampling)

1.1 词向量

What: 词组用向量表示
Why:
1.NLP转为ML问题，第一步就是将符号数学化
2词向量编码词组,使其代表N维空间中的一个点，点与点之间距离可以代表深层信息。每一个词向量的维度都可能会表征一些意义（物理含义）。例如，语义维度可以用来表明时态（过去与现在与未来），计数（单数与复数），和性别（男性与女性）
How:编码方式：比如one-hot vector

1.1.1 one-hot vector

What:对词库中n个词，每个词在某个index下取到1，其余位置为0
Disadvantge:
1.维数灾难
2.词向量无法表示词组相似性：$(W^{hotel})^T w^{motel} = (W^{hotel})^T w^{cat} = 0$[hotel和motel是近义词]。
Improve:可以把词向量的维度降低一些，在这样一个子空间中，可能原本没有关联的词就关联起来了

1.2 构造词向量方法-基于SVD

How：遍历所有的文本数据集，然后统计词出现的次数，接着用一个矩阵X来表示所有的次数情况，紧接着对X进行奇异值分解得到一个USV^T的分解。然后用U的行（rows）作为所有词表中词的词向量。对于矩阵X,有如下方法：

1.2.1 词-文档矩阵

What：行：文档M。列：词组V。
How：遍历文件，词组i出现在文件j中，将Xij值加一。得到矩阵R|V|×M

1.2.2 基于窗口的共现矩阵

What：同上，将词频换成了相关性矩阵
How: 固定大小窗口，统计每个词出现在窗口中次数。
例如： I enjoy flying. || I like NLP. || I like deep learning.

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 1: 基于窗口共现矩阵

1.2.3 奇异值分解构造词向量矩阵

What:将矩阵用更小更简单的子矩阵的相乘来表示【机器学会抽取重要特征】
Why:降维
Example:PCA、数据(图像)压缩、搜索引擎语义层次检索LSI
How:
$A = U\Sigma V^T$ 并根据保留百分比保留k个维度. 【奇异值σ跟特征值类似，在矩阵∑中也是从大到小排列，而且σ的减少特别的快，在很多情况下，前10%甚至1%的奇异值的和就占了全部的奇异值之和的99%以上了。也就是说，我们也可以用前r大的奇异值来近似描述矩阵】
Result:
U作为词嵌入矩阵，对于词表中的每一个词，都用一个k维的向量表示
Disadvantage:
1.矩阵稀疏，纬度高且经常变化
2.复杂度：$O(n^2)$(SVD).且需要对矩阵X处理

1.3 构造词向量方法-基于迭代的方法

Why：为了解决奇异值分解矩阵的问题 What：可以一步步迭代学习的模型，最终得到每个单词基于上下文的条件概率

1.3.1 语言模型（1-gram， 2-gram)

一元模型：

$$P(w_1, w_2, w_3 ... w_n) = \prod_{i=1}^{N} P(w_i)$$

二元模型：

$$P(w_1, w_2, w_3 ... w_n) = \prod_{i=1}^{N} P(w_i | w_{i-1})$$

Disadvantage:只考虑一个词语依赖相邻词语的关系

1.3.2 连续词袋模型CBOW-语言模型的进步

What：以句子其他部分为上下文，预测或产生中心词语。
How：【如下图】
1.模型已知参数：将句子表示为一些onehot向量: x(c)
2.模型输出：y(c)[其实只有一个输出]
3.模型未知参数：两矩阵：$V \in R^{n*|V|}$ 和 $U \in R^{|V|*n}$ ，n任意。
V:输入词矩阵. 词语w_i作为输入时，V的第i列v_i为w_i的输入向量
U:输出词矩阵. 词语w_j作为输出时，V的第j列v_j为w_j的输出向量
4.模型过程：
a.产生onehot向量$(x^{c-m} ... x^{c-1}, x^{c+1} ... x^{c+m})$
b.得到上下文嵌入词向量(输入):
$(v_{c-m+1} = Vx^{(c-m+1)}, ..., v_{c+m} = Vx^{(c+m)})$
c.向量取平均：$\hat{v} = \frac{2m}{v_{c-m}+v_{c-m+1}+...+v_{c+m}}$
d.得到得分向量：$z = U\hat{v}$并转化成概率分布形式$\hat{y} = softmax(z)$
e.我们希望产生的概率分布与期望的真实词语的onehot向量相匹配

如何找到U、V— 需要有一个目标函数
What:评估差异／损失的函数y与$\hat{y}$ - 交叉熵$H(\hat{y}, y) = -y_i log(\hat{y}_i)$
How: 通过如下优化函数，用梯度下降法更新每个相关词向量u_c和v_j

$$
\begin{aligned}
\text{minimize } J &= -\log P(w_c|w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}) \\
&= -\log P(u_c|\hat{v}) \\
&= -\log \frac{\exp(u_c^T\hat{v})}{\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})} \\
&= -u_c^T\hat{v} + \log\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})
\end{aligned}
$$

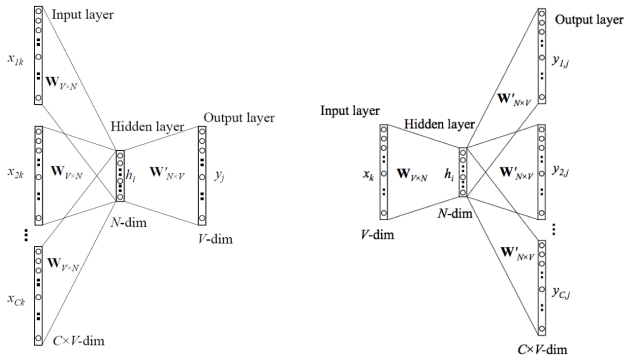

Figure 2: 左：CBOW模型过程 – 右：Skip-Gram模型过程

1.3.3 Skip-Gram模型-与CBOW相对应

What：以中心词为输入，预测周围词。则中心词为上下文
How：【如上图】
模型过程：
a.产生onehot向量x
b.得到上下文嵌入词向量(输入)：$v_c = Vx$
c.向量取平均[直接是自身]：$v_c = v_c$
d.通过u=Uv_c得到2m个得分向量：$u_{c-m},..,u_{c-1},u_{c+1},uc+m$
e.并转化成概率分布形式$y = softmax(u)$
f.我们希望产生概率分布与期望真实概率分布$y^{c-m},..,y^{c-1},y^{c+1},..,y^{c+m}$相匹配
引入朴素贝叶斯假设将联合概率拆分成独立概率相乘【给出中心词，所有输出词完全独立】，再用随机梯度下降更新未知参数。

$$
\begin{aligned}
\text{minimize } J &= -\log P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(w_{c-m+j}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(u_{c-m+j}|v_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} \frac{\exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|}\exp(u_k^T v_c)} \\
&= -\sum_{j=0,j\neq m}^{2m} u_{c-m+j}^T v_c + 2m\log\sum_{k=1}^{|V|}\exp(u_k^T v_c)
\end{aligned}
$$

1.3.4 负面抽样(Negative Samplint)-简化目标函数求值

What: 【参考文献下方】目标函数对整个单词表|V|的求和计算量巨大。要简化，则就近似。

How:对每一步训练，不去循环整个单词表，而是抽象一些负面例子。

从噪声分布$P_n(w)$中抽样，其概率分布与单词表中频率相匹配。

则只需要更新：目标函数／梯度／更新规则

负面抽样-基于SkipGram模型，但对不同目标函数优化

a.对“词-上下文”对（w, c).令P(D=1|w, c)为(w, c)来自语料库的概率

a.令P(D=0|w, c)是不来自语料库的概率

b.对P(D=1|w, c)用sigmoid函数建模：$P(D=1|w,\theta,c)=\frac{1}{1+e^{(-v_c^T v_w)}}$

c.需要新的目标函数:如果(w, c)来自语料库,目标函数能最大化P(D=1|w, c)

d.对这两个概率用最大似然 这里

$$
\begin{aligned}
\theta &= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta) \\
&= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} \log(1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)}) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})
\end{aligned}
$$

Mikolov ET AL.《Distributed Representations of Words and Phrases and their Compositionality》

2 algorithm

2.1 算法

3 machine-learning

3.1 学习算法

3.1.1 监督学习

What：数据集中每个样本都有“正确答案”，再根据样本作出预测
Example:
回归问题：
What：推导出连续的输出。
Example：房价分析／销量预测
分类问题：
What：推导出离散的输出。
Example：乳腺肿瘤判断／垃圾邮件问题

3.1.2 无监督学习

What：交给算法大量数据，让算法为我们从数据中找出某种结构
Example:
聚类问题：
How：谷歌news。同一主题的聚类

3.2 第一个学习算法-单变量线性回归

What：只有一个特征(输入变量)
回归：
What:根据之前的数据预测一个准确输出值
线性回归：
What: 确定两种或两种以上变量间相互依赖的定量关系。$y = w'x + e$
How:

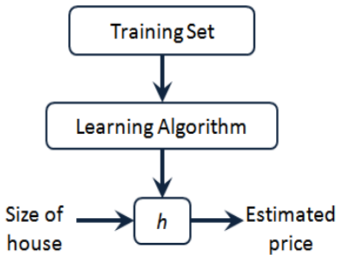

Figure 3: 监督学习算法工作方式

h的一种可能表达方式：$h_\theta(x) = \theta_0 + \theta_1 x$ 因为只有一个特征(输入变量)，这样的问题叫做单变量线性回归问题.

3.2.1 代价函数-用来求解线性回归方程参数

What:平方误差函数（平方误差代价函数）。
How:建模误差的平方和: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i)) - y^i)^2$
目标：$\text{Min}\, J(\theta_0, \theta_1)$
How：【如下图】
1. 代价函数(等高线图):在三维空间中存在一个值使得$J(\theta_0, \theta_1)$最小
2. 需要算法【自动】找出使得J 最小化的θ_0, θ_1的值

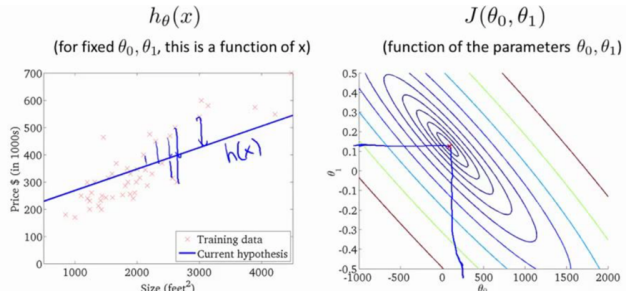

Figure 4: 代价函数-等高线图

3.2.2 梯度下降-求代价函数最小值

What:求函数最小值的算法
How：随机选择一个参数的组合($\theta_0, \theta_1, \theta_2, \theta_3$)，计算代价函数，然后寻找下一个能让代价函数值下降最多的参数组合。直到到达一个局部最小值。【由于没有常识所有的参数组合，不能保证局部最小值】